# LEARNING MONOTONE LOG-TERM DNF FORMULAS

AUTHOR(S):

Sakai, Yoshifumi; Maruoka, Akira

CITATION:

Sakai, Yoshifumi ...[et al]. LEARNING MONOTONE LOG-TERM DNF FORMULAS. 数理解析研究所講究録 1994, 871: 204-211

ISSUE DATE:

1994-05

URL:

http://hdl.handle.net/2433/84035

数理解析研究所講究録
第871巻 1994年 204-211


# LEARNING MONOTONE LOG-TERM DNF FORMULAS

酒井 義文　　丸岡 章
Yoshifumi Sakai　Akira Maruoka

*Graduate School of Information Sciences, Tohoku University*

## Abstract

Based on the uniform distribution PAC learning model, the learnability for monotone disjunctive normal form formulas with at most $O(\log n)$ terms ($O(\log n)$-term MDNF) is investigated. Using the technique of restriction, an algorithm that learns $O(\log n)$-term MDNF in polynomial time is given.

## 1 Introduction

The problem of deciding the polynomial time learnability of the class of concepts represented by disjunctive normal form formulas, denoted DNF, seems to be one of the most important and tantalizing open problems in PAC (probably approximately correct) learning paradigm introduced by Valiant[9]. Although the problem still remains open, the learnability of various classes of concepts obtained by restricting DNF in some way has been studied so far: In the distribution free setting PAC model, the class of disjunctive normal form formulas with at most $k$ literals in each term, denoted $k$-DNF, is shown to be learnable for fixed $k$[9], whereas the class of disjunctive normal form (monotone) formulas with at most $k$ terms, denoted $k$-term DNF ($k$-term MDNF), is not learnable even in the case where $k = 2$ unless RP = NP [4, 7]. If the learner is allowed to produce a formula in $k$-CNF (the class of conjunctive normal form formulas with at most $k$ literals in each clause) as hypothesis, $k$-term MDNF is shown to be learnable[7], but in this case the hypothesis can have $O(n^k)$ clauses, where $n$ is the number of variables of a target function.

There is another type of PAC model, called uniform distribution setting, where the examples are assumed to be generated according to the uniform distribution, and hence we can get more information in general about a target function from examples than in the case of distribution free setting. Since the problem of deciding the learnability of DNF remains open, it is natural to investigate the problem of the learnability for various classes in the uniform distribution setting. A few classes have been shown to be learnable in this setting: For fixed $k$, $k$-term MDNF is shown to be learnable[3, 5, 6]; The class of disjunctive normal form formulas in which each variable occurs at most once, denoted $\mu$DNF, is shown to be learnable. The latter result should be constructed with the fact that in the case of the distribution-free setting, deciding the learnability of DNF is no harder than deciding the learnability of $\mu$DNF[4]. Recently, $k$-term DNF is shown to be learnable[2] and this class is also learnable in the more relaxed distribution setting introduced in [1] where examples are assumed to be generated according to distributions deviated somehow from the uniform distribution. In the same setting of distributions, $k$-term MDNF is independently shown to be learnable using only positive examples[8], whereas the algorithm for $k$-term DNF given in [2] uses both positive and negative examples. Another related result in uniform distribution setting is given in [10], where an algorithm that learns DNF in quasi-polynomial time is proposed.

In this paper we investigate the learnability of $k$-term MDNF in the uniform distribution setting. The parameter $k$ specifying the class is allowed to depend on the number $n$ of variables of a target function.

Using the technique of restricting the examples such that appropriately chosen components of them take value 0, we deal with the case where parameter $k$ specifying the class is allowed to depend on the number $n$ of variables of a target function and give a polynomial time algorithm that learns $O(\log n)$-term MDNF in the uniform distribution setting. We adopt the strategy of restricting the examples so that all of clauses in a target function except a term are suppressed, and hence the remaining clause can be easily found. As far as a polynomial time algorithm takes the strategy, the number of clauses in MDNF that can be learned by a polynomial time algorithm must be at most $O(\log n)$.

In section 2, we illustrate the learning model and give the definition of learnability. In section 3, we give the learning algorithm for $k$-term MDNF together with an idea behind it, and in section 4 we give the proof of its correctness.

## 2 Preliminaries

Let $f$ be a Boolean function of $n$ variables $x_1, \ldots, x_n$. Given $f$, a vector $v \in \{0,1\}^n$ is called a positive example (resp. negative example) of $f$ if and only if $f(v) = 1$ (resp. $f(v) = 0$). Let $D_f^+$ (resp. $D_f^-$) denote the uniform probability distribution on $f^{-1}(1)$ (resp. $f^{-1}(0)$). Positive examples (resp. negative examples) are assumed to be generated independently according to $D_f^+$ (resp. $D_f^-$). In this paper we restrict ourselves to learning algorithms that take as input only positive examples. In order to get positive examples, a learning algorithm calls an oracle, denoted POS(), which produces positive examples independently according to $D_f^+$. So we only need the probability distribution $D_f^+$.

In the following, we often identify a Boolean formula with the Boolean function that it represents. Thus, we regard the class $F$ of Boolean formulas as the corresponding class of Boolean functions. Given a class of Boolean functions $F$, $F_n$ denotes the set of Boolean functions of $n$ variables in $F$. For each $f$ in $F$, let size($f$) denote the fewest number of symbols needed to write the representation $f$ in $F$. In the following, we write $D_f^+(f)$ to represent $\sum_{f(v)=1} D_f^+(v)$.

**Definition 1** A class of formulas $F$ is called learnable in time $t(n, \text{size}(f), \varepsilon, \delta)$ if and only if there exists a learning algorithm $L$ with oracle POS() such that for any positive integer $n$, $f \in F_n$, $\varepsilon, \delta \in (0,1)$, the algorithm $L$ halts in time $t(n, \text{size}(f), \varepsilon, \delta)$ and outputs a formula $g \in F_n$ that with probability at least $1 - \delta$ satisfies $D_f^+(f \Delta g) < \varepsilon$ and $g \subseteq f$, where $f \Delta g$ denotes the formula $(f \wedge \bar{g}) \vee (\bar{f} \wedge g)$. (Note that, in the case of $g \subseteq f$, $D_f^+(f \Delta g)$ is written as $D_f^+(f \wedge \bar{g})$.)

The formulas $f$ and $g$ in the definition above are called a target function and a hypothesis, respectively. The parameter $\varepsilon$ and $\delta$ in the definition above are called an accuracy parameter and confidence parameter, respectively.

We adopt the so called an oracle model, so the learning algorithm in Definition 1 takes as input $n$, size($f$), $\varepsilon$, and $\delta$ for a target function $f$. In general a learning algorithm is assumed to use negative examples, which are generated according to $D_f^-$, as well as positive examples, and accordingly the hypothesis $g$ it produces is required to satisfy $D_f^-(f \Delta g) < \varepsilon$ as well as $D_f^+(f \Delta g) < \varepsilon$. Since we deal with learning algorithms which use only positive examples, it turns out that we can simply replaced the condition $D_f^-(f \Delta g) < \varepsilon$ with the somewhat stronger condition $g \subseteq f$ in the definition of learnability. According to the definition we simply say $F$ is learnable instead of saying $F$ is learnable by positive examples under uniform distributions.

A literal is either a Boolean variable or its negation, and a conjunction of finite literals is called a monomial (or a term). MDNF denotes the class of monotone disjunctive normal form formulas, and $k$-term MDNF denotes the class of monotone disjunctive normal form formulas with up to $k$ terms. We use parameter $k$ as input of learning algorithm instead of size($f$) since size($f$) = $O(nk)$ for any $f$ in $k$-term MDNF. Let Term($f$) denote the set of terms of formula $f$ in MDNF. Let Var($f$) denote the set of variables that appear in formula $f$ in MDNF. In general there exist a number of formulas in MDNF that represent the same function. Throughout the paper a monotone Boolean formula is assumed to be nonredundant in the sense that a monotone formula representing a function is the one that has the fewest number of terms among such formulas so that, if $f$ is such a formula, then for any distinct terms $s$ and $t$ of $f$, $\text{Var}(s) \not\subseteq \text{Var}(t)$ holds.

Throughout the paper $\boldsymbol{v}$ denotes the random variable that takes the values in $\{v \in \{0,1\}^n \mid f(v) = 1\}$ according to the uniform distribution $D_f^+$. For $1 \leq i \leq n$, let $\boldsymbol{v}_i$ denote the random variable that takes the value of the $i$th component of $\boldsymbol{v}$. For a set $A = \{x_{i_1}, \ldots, x_{i_j}\} \subseteq \{x_1, \ldots, x_n\}$, let $\boldsymbol{v}_A$ denote $(\boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_j})$, and let $\boldsymbol{v}_A = 0$ mean that $\boldsymbol{v}_{i_1} = 0, \ldots, \boldsymbol{v}_{i_j} = 0$. For a formula $f$ in MDNF and a set $A \subseteq \{x_1, \ldots, x_n\}$, $f_A$ denotes a formula $\bigvee_{t \in \text{Term}(f), \text{Var}(t) \cap A \neq \emptyset} t$ and $f'_A$ denotes a formula $\bigvee_{t \in \text{Term}(f), \text{Var}(t) \cap A = \emptyset} t$. Later $f_{\{x_i\}}$ and $f'_{\{x_i\}}$ are simply written as $f_i$ and $f'_i$, respectively. For a set $A = \{x_{i_1}, \ldots, x_{i_j}\} \subseteq \{x_1, \ldots, x_n\}$, we say that the set $A$ suppresses the terms in Term($f_A$). This is because putting $x_{i_1} = 0, \ldots, x_{i_j} = 0$ makes all of the terms in Term($f_A$) 0. Natural logarithm is written as "ln". The cardinality of a set $S$ is denoted $|S|$.

Before closing this section, we give a lemma due to [6] that will be used in the following sections. Procedure FREQ given in Figure 1 can be used to estimate the conditional probability of event $E(\boldsymbol{v})$ under condition $C(\boldsymbol{v})$ from the fraction of its occurrence in a sequence of trials. As in the definition of learnability, $\delta$ is called confidence parameter of FREQ.

**Lemma 2.1** ([6]) *Let $p$, $q$ and $\delta$ be such that $0 < p \leq 1$, $0 < q \leq \frac{2}{3}p$ and $0 < \delta < 1$. If $\Pr[E(\boldsymbol{v}) \mid C(\boldsymbol{v})] \leq p - q$ then* FREQ *returns "high" with probability at most $\delta$, and if $\Pr[E(\boldsymbol{v}) \mid C(\boldsymbol{v})] \geq p$ then* FREQ *returns "low" with probability at most $\delta$.*

**procedure** FREQ$(p-q,\ p,\ E(\boldsymbol{v}),\ C(\boldsymbol{v}),\ \delta)$:
**begin**
  $r \leftarrow (12/q^2)\ln(1/\delta)$;
  call POS() repeatedly until $r$ positive examples that satisfy $C(\boldsymbol{v})$ are found;
  **if** $E(\boldsymbol{v})$ holds at least $r(p-q/2)$ these positive examples **then**
    **return** "high"
  **else return** "low"
**end**

Figure 1: Procedure FREQ

**Algorithm** LEARN
**input:** $n,\ k,\ \varepsilon,\ \delta$
**output:** $g$
**begin**
  $d \leftarrow 2(k+3nk^2+2k^3)$;
  $g \leftarrow \emptyset$;
  **while** FREQ$(\varepsilon/2,\ \varepsilon,\ g(\boldsymbol{v})=0,$ **true**, $\delta/d)$ = "high" **do**
    **begin**
      $t \leftarrow$ MAKE_TERM$(g)$;
      $g \leftarrow g \vee t$
    **end**
**end**

Figure 2: Algorithm LEARN

# 3 Results

The main result of the present paper is stated as the next theorem.

**Theorem 3.1** *The class* $k$-term MDNF *is learnable in time polynomial in* $n,\ k,\ 2^k,\ 1/\varepsilon$ *and* $\ln(1/\delta)$.

The above theorem immediately implies the following corollary.

**Corollary 3.2** *For* $k = O(\log n)$, *the class* $k$-term MDNF *is learnable in time polynomial in* $n,\ 1/\varepsilon$ *and* $\ln(1/\delta)$.

We shall verify the theorem by giving a learning algorithm which is based on the notion of restriction given by [6]. We first give an outline of the algorithm together with an idea behind it.

Let $f$ be a target in $k$-term MDNF, and let $\boldsymbol{v}$ be a random positive example of $f$. The algorithm proceeds finding terms of $f$ one by one. Suppose that the algorithm succeeded to find some terms in Term($f$) and that the disjunction of these terms is denoted $g$. The algorithm continues to find one of the remaining terms in Term($f$) − Term($g$) by calling procedure MAKE_TERM given in Figure 3 until it finds almost all of the remaining terms.

In order to find the remaining terms, we employ the technique of restricting the domain of a target function to a smaller region: We restrict ourselves to positive examples $\boldsymbol{v}$ such that $\boldsymbol{v}_A = 0$ for $A$ chosen appropriately so that we may consider as a target function $f'_A$ with $n-|A|$ variables rather than $f$ by ignoring the variables corresponding to the elements in $A$. This is because $A$ suppresses all the terms in $f_A$. If we can find $A$ such that $f'_A$ contains only one term, then we can easily find the term by observing sufficiently many positive vectors $\boldsymbol{v}$ such that $\boldsymbol{v}_A = 0$. This is because with high probability it holds that for sufficiently many positive vectors $\boldsymbol{v}$ with $\boldsymbol{v}_A = 0$ the components of vectors $\boldsymbol{v}$ take value 1 if and only if the components correspond to the variables of the single term not suppressed by $A$. So all we have to do is to find such $A$. To do this we provide two stages: The first stage (step 1 in procedure MAKE_TERM) produces $A$ that suppresses $g$; By

**procedure** MAKE_TERM($g$):
**begin**
step 1: $A \leftarrow \emptyset$;
**while** Term($g'_A$) $\neq \emptyset$ **do** (* $g'_A$ is formula $\bigvee_{t \in \text{Term}(g), \text{Var}(t) \cap A = \emptyset} t$ *)
**begin**
$i \leftarrow$ CHOOSE(Var($g'_A$), $g(v) = 0$ and $v_A = 0$);
$A \leftarrow A \cup \{x_i\}$
**end**;
step 2: **repeat forever**
**begin**
let $T$ be a subset of $\{x_1, \ldots, x_n\} - A$
consisting of such $x_j$'s in $\{x_1, \ldots, x_n\} - A$ that
FREQ $\left(\frac{1}{2} - \frac{1}{2^{k+1}k^2}, \frac{1}{2}, v_j = 0, v_A = 0, \frac{\delta}{d}\right)$ = "low";
$i \leftarrow$ CHOOSE($T$, $v_A = 0$);
**if** $i = 0$ **then return** $\bigwedge_{x_j \in T} x_j$;
$A \leftarrow A \cup \{x_i\}$
**end**
**end**

**procedure** CHOOSE($V$, $C(v)$):
**begin**
$l \leftarrow k$;
$i \leftarrow 0$;
**for each** $x_j$ in $V$ **do**
**if** $l > 0$ and FREQ $\left(\frac{1}{2(l+1)}, \frac{1}{2l}, v_j = 0, C(v), \frac{\delta}{d}\right)$ = "high" **then**
**begin**
$i \leftarrow j$;
**repeat**
$l \leftarrow l - 1$
**until** $l = 0$ or FREQ $\left(\frac{1}{2(l+1)}, \frac{1}{2l}, v_j = 0, C(v), \frac{\delta}{d}\right)$ = "low"
**end**;
**return** $i$
**end**

Figure 3: Procedures MAKE_TERM and CHOOSE

adding appropriate variables to A the second stage (step 2 in procedure MAKE_TERM) yields $A_s$ such that $f'_{A_s}$ consists of a single term. These stages find $A$ by calling repeatedly procedure CHOOSE which will be described below. In finding set $A$ in these stages, oracle POS() is called until vector $v$ such that $v_A = 0$ is found. So it will take much time to find such $v$ if the probability $\Pr[v_A = 0]$ is small. To make the algorithm to halt in polynomial time, we are required to choose $A$ such that $\Pr[v_A = 0]$ is large and that $A$ suppresses a large number of terms in $f$. As we will see in the proof of Fact 4.8 there is a sort of tradeoff between the two requirements. This is the critical point of the argument that will be dealt with later in detail. This is a rough sketch of the whole algorithm. Strictly speaking, the algorithm sometimes works in a somewhat different way from what we mentioned. This is because the algorithm is constructed based on procedure FREQ which yields an output with some tolerance: From an answer of procedure FREQ$(p-q,\ p,\ E(v),\ C(v),\ \delta)$, we can conclude with probability $1-\delta$ which of $\Pr[E(v) \mid C(v)] \geq p-q$ and $\Pr[E(v) \mid C(v)] \leq p$ holds. In other words, an answer gives us no information when $p-q \leq \Pr[E(v) \mid C(v)] \leq p$ holds.

We now explain in more detail how to find the variable set $A$ such that $f'_A$ consists of a single term. Step 1 and 2 in procedure MAKE_TERM do this by calling procedure CHOOSE to find a new variable and repeatedly adding it to the set $A$ previously found. When procedure CHOOSE$(V, C(v))$ is called, $V$ is taken to be the set of the possible variables added and $C(v)$ is taken to be the condition to restrict the domain appropriately. Procedure CHOOSE$(V, C(v))$ first checks if $\Pr[v_j = 0 \mid C(v)] \leq 1/(2k)$ holds for all $x_j$ in $V$. If it is so, then it returns 0, which indicates that $V$ constitutes the single term of $f'_A$. Roughly speaking, this is because, when $C(v)$ is taken to be $v_A = 0$, the condition says that $\Pr[v_j = 1 \mid v_A = 0] \geq 1 - 1/(2k)$ holds for all $x_j$'s in $V$, which means that the components, corresponding to $V$, of a positive vector $v$ with $v_A = 0$ probably has value 1 (Fact 4.4). Otherwise it finds $x_i$ that maximizes $\Pr[v_i = 0 \mid C(v)]$ among the variables in $V$ and returns $i$ ($\neq 0$). Because newly added variable $x_i$ is chosen in this way, it is guaranteed that for variable set $A$ finally obtained $\Pr[v_A = 0]$ becomes large. Let $g$ be the disjunction of the terms and $A$ be the set both found by the algorithm LEARN so far. Step 1 in MAKE_TERM calls CHOOSE(Var($g'_A$), $g(v) = 0$ and $v_A = 0$) and adds the returned variable to $A$ to make the new variable set until all of the terms in $g$ are suppressed by $A$. Then step 1 passes the set $A$ chosen to step 2. Step 2 makes the variable set $T$ consisting of all the variables $x_j$ in $\{x_1, \ldots, x_n\} - A$ such that $\Pr[v_j = 0 \mid v_A = 0] < 1/2$, and then calls CHOOSE($T$, $v_A = 0$), and finally adds the returned variable to $A$ repeatedly until all of the variables $x_j$ in $T$ satisfies $\Pr[v_j = 0 \mid v_A = 0] \leq 1/(2k)$, which means that $T$ constitutes a single term in Term($f$) − Term($g$). We notice that, in order to choose set $T$ of the possible variables to restrict the domain, we use the condition $\Pr[v_j = 0 \mid v_A = 0] < 1/2$ because we need to prevent a variable $x_j$ which does not appear in any remaining terms from being selected as a variable in $T$ (For such a variable $x_j$, clearly $\Pr[v_j = 0 \mid v_A = 0] = 1/2$ holds).

# 4 Correctness

In this section, we show that Algorithm LEARN in Figure 2 learns $k$-term MDNF in time polynomial in $n$, $k$, $2^k$, $1/\varepsilon$ and $\ln(1/\delta)$ by verifying Lemma 4.1 and Lemma 4.6. The former shows the correctness of the algorithm, while the latter estimates the time complexity of the algorithm.

Many of the statements in this section are involved with Lemma 2.1, so we simply claim that the fact that FREQ returns "low" implies $\Pr[E(v) \mid C(v)] < p$, and that the fact that FREQ returns "high" implies $\Pr[E(v) \mid C(v)] > p - q$ without saying the phrase "with high probability". The probability that this is not the case turns out to be sufficiently small because we take the confidence parameter sufficiently small.

Because of FREQ built in the algorithm LEARN, LEARN is not guaranteed to halt in polynomial time. But for simplicity we use the learning algorithm shown in Figures 2 and 3 for the time being. Later, as it will be mentioned in the proof of Lemma 4.6, we shall modify the algorithm somehow. The algorithm is forced to halt in polynomial time at an appropriate time. It turns out that the probability of this happening is so small that it does not violate the condition of the learnability.

**Lemma 4.1** *Let $f$ be a target function in $k$-*term MDNF. *Algorithm* LEARN *outputs a formula $g$ in $k$-*term MDNF *that with probability at least $1-\delta/2$ satisfies $D_f^+(f \Delta g) < \varepsilon$ and $g \subseteq f$.*

To prove Lemma 4.1, we need a series of facts.

**Fact 4.2** *Let $V$ be a subset of the set $\{x_1, \ldots, x_n\}$, and let $l$ be the value of $l$ at the execution of return statement in the procedure* CHOOSE *called with parameters $V$ and $C(v)$. If* CHOOSE *returns* 0 *then for any variable $x_j$ in $V$, $\Pr[v_j = 0 \mid C(v)] < 1/(2k)$ holds, and if* CHOOSE *returns $i$ not equal to* 0 *then $\Pr[v_i = 0 \mid C(v)] > 1/(2(l+2))$ holds and for any variable $x_j$ in $V$, $\Pr[v_j = 0 \mid C(v)] < 1/(2l)$ holds.*

**Fact 4.3 ([6])** *Let $f$ be in $k$-term* MDNF *and $t$ be any term in* $\mathrm{Term}(f)$. *For any $x_i$ in* $\mathrm{Var}(t)$, $\Pr[v_i = 0] \leq 1/2 - (1/2^k)\Pr[t(v) = 1]$.

**Fact 4.4** *Let $f$ be a target function. If algorithm* LEARN *calls the procedure* MAKE_TERM *with parameter $g$ such that* $\mathrm{Term}(g) \subseteq \mathrm{Term}(f)$, *then* MAKE_TERM *returns a term in* $\mathrm{Term}(f) - \mathrm{Term}(g)$.

**Proof:** Let $A$ and $T$ be those at the execution of the return statement in step 2. Since $A$ suppresses all terms in $\mathrm{Term}(g)$ and there remain at most $k$ terms not being suppressed by $A$, there exists term $s$ in $\mathrm{Term}(f) - \mathrm{Term}(g)$ such that $\Pr[s(v) = 1 \mid v_A = 0] \geq 1/k$. We therefore have by Fact 4.3 that for any $x_j$ in $\mathrm{Var}(s)$ $\Pr[v_j = 0 \mid v_A = 0] \leq 1/2 - 1/(2^k k)$, which implies $\Pr[v_j = 0 \mid v_A = 0] < 1/2 - 1/(2^{k+1}k^2)$. Hence by the condition of $T$ we have $\mathrm{Var}(s) \subseteq T$. Assume in contradiction that $\mathrm{Var}(s) \subsetneq T$ holds. Let $x_j$ be a variable in $T - \mathrm{Var}(s)$. We have $\Pr[v_j = 0 \text{ and } s(v) = 1 \mid v_A = 0] = \Pr[v_j = 0 \mid s(v) = 1 \text{ and } v_A = 0] \cdot \Pr[s(v) = 1 \mid v_A = 0] \geq (1/2) \cdot (1/k)$, which implies $\Pr[v_j = 0 \mid v_A = 0] \geq 1/(2k)$. This is contradiction because, when MAKE_TERM returns formula $\bigwedge_{x_l \in T} x_l$ $i$ takes value 0, and hence for any $x_l$ in $T$ $\Pr[v_l = 0 \mid v_A = 0] < 1/(2k)$. □

**Fact 4.5** *Let $f$, $g$ and $h$ be formulas in* MDNF *such that $f = g \vee h$,* $\mathrm{Term}(g) \neq \emptyset$ *and* $\mathrm{Term}(h) \neq \emptyset$. *There exists a variable $x_i$ in* $\mathrm{Var}(g)$ *such that* $\Pr[v_i = 0 \mid g(v) = 0] \geq 1/(2|\mathrm{Term}(h)|)$.

**Proof:** There exists a term $t$ in $\mathrm{Term}(h)$ such that $\Pr[t(v) = 1 \mid g(v) = 0] \geq 1/|\mathrm{Term}(h)|$. Then, for any variable $x_i$ in $\mathrm{Var}(g) - \mathrm{Var}(t)$, $\Pr[v_i = 0 \mid g(v) = 0] \geq (1/2)\Pr[t(v) = 1 \mid g(v) = 0] \geq 1/(2|\mathrm{Term}(h)|)$. Since any formula is assumed to be nonredundant throughout the paper, $\mathrm{Var}(g) - \mathrm{Var}(t)$ is not empty. Thus it completes the proof. □

We now proceed to the proof of Lemma 4.1.

**Proof of Lemma 4.1:** Since the correctness of the algorithm is immediate from Fact 4.4, we only need to estimate the confidence parameters.

Note that the number of iteration of the while statement in step 1 in MAKE_TERM is at most $|\mathrm{Term}(g)| \leq k$, since by Fact 4.2 and 4.5 the value of $i$ returned by CHOOSE is not 0 throughout the execution of the while statement, and each variable $x_i$ added to $A$ in step 1 suppresses at least one term in $\mathrm{Term}(g)$. Also note that the number of iteration of repeat statement in step 2 is at most $|\mathrm{Term}(f)| - |\mathrm{Term}(g)| \leq k$, since each variable $x_i$ added to $A$ is chosen from $T$, which implies that $\Pr[v_i = 0 \mid v_A = 0] < 1/2$ and hence the variable $x_i$ suppresses at least one term in $\mathrm{Term}(f)$ not suppressed by $A$.

The algorithm LEARN calls at most $k$ times MAKE_TERM together with FREQ. Step 1 in MAKE_TERM calls CHOOSE at most $k$ times, while step 2 in MAKE_TERM calls CHOOSE at most $k$ times together with FREQ being called at most $n$ times for each call for CHOOSE. On the other hand, CHOOSE calls FREQ at most $n + k$ times. Summing up these number of calls for FREQ, we conclude that the total number of calls for FREQ in the whole algorithm is at most $k + k(k(n + k) + k(2n + k)) = k + 3nk^2 + 2k^3$. Putting $d = 2(k + 3nk^2 + 2k^3)$, we can conclude that the probability that all of the FREQ called in the whole algorithm give the right answers in the sense described at the begining of this section is bounded below by $(1 - \delta/d)^{d/2} \geq 1 - \delta/2$. Thus, since the condition of the while statement in Algorithm LEARN guarantees that $\Pr[g(v) = 0] < \varepsilon$, i.e., $D_f^+(f \Delta g) < \varepsilon$, the lemma follows. □

**Lemma 4.6** *Algorithm* LEARN *halts in time polynomial in $n$, $k$, $2^k$, $1/\varepsilon$ and* $\ln(1/\delta)$.

To prove Lemma 4.6, we also need another series of facts.

**Fact 4.7** *Let $f$, $g$ and $h$ be formulas in* MDNF *such that $f = g \vee h$ and* $\mathrm{Term}(h) \neq \emptyset$, *let $T$ be a subset of the set $\{x_1, \ldots, x_n\}$ such that $T - \mathrm{Var}(t) \neq \emptyset$ for any term $t$ in* $\mathrm{Term}(h)$, *and let $l \geq 2$. If there exists a variable $x_i$ in $T$ such that* $|\mathrm{Term}(h_i)| \leq l - 1$, *then there exists a variable $x_j$ in $T$ such that* $\Pr[v_j = 0 \mid g(v) = 0] \geq 1/(2l)$.

**Proof:** Let variable $x_i$ be as in the fact. If $\Pr[v_i = 0 \mid g(v) = 0] \geq 1/(2l)$, then the fact holds. So we assume $\Pr[v_i = 0 \mid g(v) = 0] < 1/(2l)$, or equivalently $\Pr[v_i = 0 \text{ and } g(v) = 0] < \Pr[g(v) = 0]/(2l)$.

Under the condition that $g(v) = 0$, $h_i(v) = 0$ implies $h_i'(v) = 1$. We therefore have $\Pr[g(v) = 0 \text{ and } h_i(v) = 1] = \Pr[g(v) = 0] - \Pr[g(v) = 0 \text{ and } h_i(v) = 0] \geq \Pr[g(v) = 0] - \Pr[g(v) = 0 \text{ and } h_i'(v) = 1]$. On the other hand, it is easy to see we have $\Pr[g(v) = 0 \text{ and } h_i'(v) = 1] \leq 2\Pr[v_i = 0,\ g(v) = 0 \text{ and } h_i'(v) = 1] < 2\Pr[g(v) = 0]/(2l)$. This is because the second inequality follows from the assumption, whereas the first inequality follows from the fact that, if $v$ with $v_i = 1$ satisfies both $g(v) = 0$ and $h_i'(v) = 1$, then the vector obtained from $v$ by changing the $i$th component from 1 to 0 also satisfies the same condition. Hence from these inequalities we have $\Pr[g(v) = 0 \text{ and } h_i(v) = 1] > (1 - 1/l)\Pr[g(v) = 0]$. On the other hand, it

follows from $|\mathrm{Term}(h_i)| < l-1$ that there exists a term $t$ in $\mathrm{Term}(h_i)$ such that $\Pr[g(v)=0 \text{ and } t(v)=1] \geq \Pr[g(v)=0 \text{ and } h_i(v)=1]/|\mathrm{Term}(h_i)| \geq (1-1/l)\Pr[g(v)=0]/(l-1) = (1/l)\Pr[g(v)=0]$. Thus, for any $x_j$ in $T - \mathrm{Var}(t)$, we have $\Pr[v_j=0 \text{ and } g(v)=0] \geq \Pr[v_j=0,\ t(v)=1 \text{ and } g(v)=0] \geq (1/2)\Pr[t(v)=1 \text{ and } g(v)=0] \geq (1/(2l))\Pr[g(v)=0]$, which implies $\Pr[v_j=0 \mid g(v)=0] \geq 1/(2l)$. This completes the proof. □

**Fact 4.8** *Let $f$ be a target function and let $A$ be the set chosen in step 1 of procedure* MAKE_TERM *with parameter $g$ in* MDNF *such that* $\mathrm{Term}(g) \subsetneq \mathrm{Term}(f)$. *Then* $\Pr[v_A=0] > (1/2^{3k-1})\Pr[g(v)=0]$.

**Proof:** Let $A_r$, $i_r$ and $l_r$ denote the values of $A$, $i$ and $l$, respectively, just after the $r$th execution of CHOOSE in step 1 in procedure MAKE_TERM. Let $A$ be passed from step 1 to step 2 after the $m$th iteration in step 1 so that $A = A_m \cup \{x_{i_m}\} = \{x_{i_1}, \ldots, x_{i_m}\}$. In particular, let $A_{m+1} = A$. We first note that CHOOSE in step 1 does not return 0 as the value of $i$ throughout the iteration. This is because by Fact 4.5 for any $1 \leq i \leq m$ there exists $x_j$ in $\mathrm{Var}(g'_{A_r})$ such that $\Pr[v_j=0 \mid g'_{A_r}(v)=0 \text{ and } v_{A_r}=0] \geq 1/(2k)$, hence the first FREQ in CHOOSE does not return "low" for the variable. Let $h$ be a formula in MDNF such that $h \vee g = f$. By Fact 4.2, $\Pr[v_j=0 \mid g'_{A_r}(v)=0 \text{ and } v_{A_r}=0] < 1/(2l_r)$ holds for any $x_j$ in $\mathrm{Var}(g'_{A_r})$ and any $1 \leq r \leq m$, which in turn, together with Fact 4.7, implies $|\mathrm{Term}((h'_{A_r})_{i_r})| = |\mathrm{Term}(h'_{A_r})| - |\mathrm{Term}(h'_{A_{r+1}})| > l_r - 1$ since $\mathrm{Var}(g'_{A_r}) - \mathrm{Var}(t) \neq \emptyset$ for any term $t$ in $\mathrm{Term}(h)$. Again by Fact 4.2, we have $\Pr[v_{i_r}=0 \mid g'_{A_r}(v)=0 \text{ and } v_{A_r}=0] > 1/(2(l_r+2))$. Since $v_A=0$ implies $g(v)=0$, we have by the inequalities above

$$
\begin{aligned}
\Pr[v_A=0] &= \Pr[g(v)=0 \text{ and } v_A=0] \\
&= \Pr[g(v)=0]\Pr[v_A=0 \mid g(v)=0] \\
&= \Pr[g(v)=0] \cdot \prod_{r=1}^{m} \Pr[v_{i_r}=0 \mid g'_{A_r}(v)=0 \text{ and } v_{A_r}=0] \\
&> \Pr[g(v)=0] \cdot \prod_{r=1}^{m} \frac{1}{2(l_r+2)} \\
&> \Pr[g(v)=0] \cdot \left( \prod_{r=1}^{m} \left( 2(|\mathrm{Term}(h'_{A_r})| - |\mathrm{Term}(h'_{A_{r+1}})| + 3) \right) \right)^{-1} \\
&\geq \Pr[g(v)=0] \cdot \left( 2^m \cdot 3^m \cdot 4^{|\mathrm{Term}(h)|} \right)^{-1}.
\end{aligned}
$$

Putting $d_r = |\mathrm{Term}(h'_{A_r})| - |\mathrm{Term}(h'_{A_{r+1}})|$ and $d = |\mathrm{Term}(h)|$, the last inequality follows from the fact that, for any nonnegative integers $d_r$'s such that $\sum_{r=1}^{m} d_r \leq d$, $\prod_{r=1}^{m}(d_r+3) \leq 3^m \cdot 4^d$ holds. □

**Fact 4.9** *Let $f$ be a target function and let $A$ be the set chosen in step 2 of procedure* MAKE_TERM *with parameter $g$ in* MDNF *such that* $\mathrm{Term}(g) \subsetneq \mathrm{Term}(f)$. *Then* $\Pr[v_A=0] > (1/2^{8k-1})\Pr[g(v)=0]$.

**Proof:** Let $A_r$, $T_r$, $i_r$ and $l_r$ denote the values of $A$, $T$, $i$ and $l$, respectively, just after the $r$th execution of CHOOSE in step 2 in procedure MAKE_TERM. Assume that CHOOSE called in step 2 return 0 as the value of $i$ for the first time in the $m$th iteration in step 2 so that $i_m = 0$ and for any $1 \leq r < m$, $i_r \neq 0$.

We first show the claim that for any $1 \leq r < m$, there exists a term $s$ in $\mathrm{Term}(f'_{A_r})$ such that $\mathrm{Var}(s) \subsetneq T_r$ which implies that $T_r - \mathrm{Var}(t) \neq \emptyset$ for any term $t$ in $\mathrm{Term}(f'_{A_r})$. There exists a term $s$ in $\mathrm{Term}(f'_{A_r})$ such that $\Pr[s(v)=1 \mid v_{A_r}=0] \geq 1/|\mathrm{Term}(f'_{A_r})| \geq 1/k \geq 1/(2k^2)$. By Fact 4.3 and the fact that in the $r$th execution any variable $x_j$ in $\{x_1, \ldots, x_n\} - A_{r-1}$ such that $\Pr[v_j=0 \mid v_{A_{r-1}}=0] \leq 1/2 - 1/(2^{k+1}k^2)$ is taken to be in $T_r$, we have $\mathrm{Var}(s) \subseteq T_r$ for any $1 \leq r < m$. We show that for this term $s$, $\mathrm{Var}(s) \neq T_r$ holds. Assume in contradiction that $\mathrm{Var}(s) = T_r$. For any variable $x_j$ in $\{x_1, \ldots, x_n\} - (T_r \cup A_r)$, $\Pr[v_j=0 \mid v_{A_r}=0] > 1/2 - 1/(2^{k+1}k^2)$, which in turn, together with Fact 4.3, implies that for any term $t$ in $\mathrm{Term}(f'_{A_r}) - \{s\}$, $\Pr[t(v)=1 \mid v_{A_r}=0] < 1/(2k^2)$. Since there exist at most $k$ terms in $\mathrm{Term}(f'_{A_r}) - \{s\}$, we have therefore that for any variable $x_i$ in $T_r$, $\Pr[v_i=0 \mid v_{A_r}=0] \leq (1/2)\sum_{t \in \mathrm{Term}(f'_{A_r})-\{s\}} \Pr[t(v)=1 \mid v_{A_r}=0] < (1/2) \cdot k \cdot 1/(2k^2) = 1/(4k) \leq 1/(2(k+1))$, hence CHOOSE returns 0 as the value of $i_r$, contradicting the assumption. Thus the claim is verified.

By Fact 4.2, for any $1 \leq r < m$ and any $x_j$ in $T_r$, $\Pr[v_j=0 \mid v_{A_r}=0] < 1/(2l_r)$ holds. By the above claims, the inequalities and Fact 4.7, we have for any $1 \leq r < m$, $|\mathrm{Term}((f'_{A_r})_{i_r})| > l_r - 1$, which implies $|\mathrm{Term}(f'_{A_{r+1}})| < |\mathrm{Term}(f'_{A_r})| - l_r + 1$. By a similar argument to the proof of Fact 4.8, we have $\Pr[v_A=0] > (1/2^{5k}) \cdot \Pr[v_{A_1}=0]$. This is because $\sum_{r=1}^{m-1}(|\mathrm{Term}(f'_{A_r})| - |\mathrm{Term}(f'_{A_{r+1}})|) \leq |\mathrm{Term}(f)|$ and

$m-1 \leq |\mathrm{Term}(f)|$. On the other hand, by Fact 4.8, $\Pr[v_{A_1}=0] > (1/2^{3k-1})\Pr[g(v)=0]$ holds. Thus we have $\Pr[v_A=0] > (1/2^{5k})\cdot\Pr[v_{A_1}=0] > (1/2^{5k})(1/2^{3k-1})\cdot\Pr[g(v)=0] = (1/2^{8k-1})\cdot\Pr[g(v)=0]$, establishing the fact. □

Noting that the while loop in the learning algorithm guarantees that $\Pr[g(v)=0] > \varepsilon/2$, we have the next fact from Fact 4.8 and Fact 4.9.

**Fact 4.10** *Given parameter $g$ with* $\mathrm{Term}(g) \subsetneq \mathrm{Term}(f)$, $\Pr[g(v)=0 \text{ and } v_A=0] > \varepsilon/2^{8k}$ *holds throughout procedure* MAKE_TERM.

**Fact 4.11** ([9]) *The probability of at most $r$ successes in at least $(2/p)(r+\ln(1/\delta))$ independent trials with probability of success at least $p$ is at most $\delta$.*

We now proceed to the proof of Lemma 4.6 which, together with Lemma 4.1, completes the proof of Theorem 3.1.

**Proof of Lemma 4.6:** In order to make algorithm LEARN halt in polynomial time, we have to modify FREQ slightly. In view of Fact 4.11, the probability that at most $r$ positive examples satisfying $C(v)$ are found while FREQ calls POS() $(2/\Pr[C(v)])(r+\ln(d/\delta))$ times is at most $\delta/d$. So if we modify FREQ by replacing the call statement in the procedure by "call POS() repeatedly until $(2/\Pr[C(v)])(r+\ln(d/\delta))$ times or $r$ positive examples satisfying $C(v)$ are found", the probability that the modified FREQ is forced to stop because of the time limit is at most $\delta/d$. So in the same argument as that of the proof of Lemma 4.1, the probability that FREQ is forced to stop during the whole algorithm is at most $\delta/2$.

We now estimate the time complexity of FREQ$(p-q,\ p,\ E(v),\ C(v),\ \delta/d)$ which is called by LEARN. It is easy to see that the time complexity is given by $O((2/\Pr[C(v)])(r+\ln(d/\delta)))$. Now we have to estimate each factor in the quantity. First of all we can simply put $d = 2(k+3nk^2+2k^3)$ and $r = (12/q^2)\ln(1/\delta)$. Moreover it is easy to see that the value of $1/\Pr[C(v)]$ in each FREQ is at most $2^{8k}/\varepsilon$ and that the value of $1/q$ in each FREQ is at most $\max\{2/\varepsilon, 2^{k+1}k^2\}$. Thus we can see the time complexity of FREQ is polynomial in $k$, $2^k$, $1/\varepsilon$ and $\ln(n/\delta)$. Finally, it is not hard to see that the time complexity of LEARN is given as in the lemma. □

# References


[1] P. L. Bartlett, R. C. Williamson, Investigating the distribution assumptions in the PAC learning model, in *Proceedings of the 4th Annual Workshop on Computational Learning Theory*, Morgan Kaufmann, 1991.

[2] M. Flammini, A. Marchetti-Spaccamela and L. Kucera, Learning DNF Formulae under Classes of Probability Distributions, in *Proceedings of the 5th Annual Workshop on Computational Learning Theory*, Morgan Kaufmann, 1992, pp.85–92.

[3] Q. P. Gu and A. Maruoka, Learning Monotone Boolean Functions by Uniformly Distributed Examples, *SIAM Journal on Computing*, Vol.21, No.3, 1992, pp.587–599.

[4] M. Kearns, M. Li, L. Pitt and L. G. Valiant, On the Learnability of Boolean Formulae, in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, 1987, pp.285–295.

[5] L. Kucera, A. Marchetti-Spaccamela and M. Protasi, On the learnability of DNF formulae, Lecture Notes in *Computer Science*, Vol.**317**, 1988, pp.347–361.

[6] T. Ohguro and A. Maruoka, A learning algorithm for monotone $k$-term DNF, in *Proceedings of FUJITSU IIAS-SIS Workshop on Computational Learning Theory*, 1989.

[7] L. Pitt and L. G. Valiant, Computational limitation on learning from examples, *Journal of the ACM*, Vol.**35**, No.4, 1988, pp.965–984.

[8] Y. Sakai and A. Maruoka, Laerning $k$-term Monotone Boolean Formulae, in *Proceedings of the 3rd Workshop on Algorithmic Learning Theory*, 1992, pp.197–207.

[9] L. G. Valiant, A theory of the learnable, *Communications of the ACM*, **27**(11), 1984, pp.1134–1142.

[10] K. Verbeurgt, Learning DNF under the Uniform Distribution in Quasii-Polynomial Time, in *Proceedings of the 3rd Annual Workshop on Computational Learning Theory*, Morgan Kaufmann, 1990, pp.314–326.